HMS Calorimeter in HCANA

Recent progress

Vardan Tadevosyan
Simon Zhamkochyan

Hall A/C software meeting
14 Jan 2015
Calibration replay

- h_cal_replay.cpp (<RunNumber, NumberOfEvents>)

  replays a run, generates ntuple with raw data

  Similar to hodtest.C and hodtest_mkj.C, with minor modifications like cut and output files, MaxEventToReplay etc.

  - Works on single track electron events
  - Selects electrons in Gas Cherenkov
  - Also makes use of $\beta_{TOF}$ for $e^-$ selection
## Calibration replay

Event selection done in `hcal_replay_cuts.def`:

**Block: Reconstruct**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>one_track</td>
<td>H.tr.n==1</td>
</tr>
<tr>
<td>one_clust</td>
<td>H.cal.nclust==1</td>
</tr>
<tr>
<td>one_sh_track</td>
<td>H.cal.ntracks==1</td>
</tr>
<tr>
<td>in_delta</td>
<td>H.tr.tg_dp&gt;-10.&amp;&amp;H.tr.tg_dp&lt;10.</td>
</tr>
<tr>
<td>good_cer</td>
<td>H.cer.npesum&gt;3.</td>
</tr>
<tr>
<td>good_beta</td>
<td>H.tr.beta&gt;0.740&amp;&amp;H.tr.beta&lt;0.935</td>
</tr>
</tbody>
</table>

# This version is for calibration from scratch (no calibration constants exist, # first time calibration).

```plaintext
Reconstruct_master one_track && one_clust && in_delta && good_cer && good_beta
```

# This version can be used for iterative calibration (improve existing constants).

```plaintext
#Reconstruct_master one_track && one_sh_track && in_delta && good_cer &&
#good_beta
```
Calibration replay

Ntuple selection done in `output_hcal_replay.def`:

```plaintext
#
# Output definition for the HMS calorimeter calibration.
#
block H.cal.* ← comprises pedestal subtracted ADC signals
variable H.cer.npesum
variable H.tr.beta
variable H.tr.p
variable H.tr.tg_dp
variable H.tr.y  # Y FP
variable H.tr.x  # X FP
variable H.tr.ph # tan(phi), wrt Y axis
variable H.tr.th # tan(theta), wrt X axis
variable H.tr.n  # number of tracks
```
Calibration Algorithm

Definitions:

\[ q = \begin{pmatrix} q_1 \\ \vdots \\ q_N \end{pmatrix} \text{ -- QDC signals}, \quad \alpha = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} \text{ -- calibration constants}, \]

\[ e_0 = E(e) \text{ -- projectile mean energy}, \quad q_0 = E(q) \text{ -- mean signals}, \]

\[ e_R = \alpha^T q \text{ -- reconstructed energy}. \]

Constrained minimization: Minimize variance of the reconstructed energy with respect to the projectile energy

Minimize \( E(e_R - e)^2 \) subject to \( \alpha^T q_0 = e_0 \)

\[ \alpha_C = \frac{e_0 - \alpha_U^T q_0}{q_0^T Q^{-1} q_0} Q^{-1} q_0 + \alpha_U \]

\[ Q = E(qq^T), \quad \alpha_U = Q^{-1} q_e, \quad q_e = E(eq) \]

In `hcal_calib.cpp`:

```cpp
#include "THcShowerCalib.h" ← the Shower calibration class

... theShowerCalib.Init(); // Initialize constants and variables, histogram declaration

theShowerCalib.CalcThresholds();
Calculate Thresholds on the uncalibrated Edep/P. These are used mainly to exclude potential hadronic events due to the Gas Cherenkov inefficiency.
• Build energy distribution using uncalibrated gains: 0.5 for the first 2 columns, 1 for others
• Set thresholds as ±3*RMS

theShowerCalib.ComposeVMs(); // Fill in vectors and matrices

theShowerCalib.SolveAlphas(); // Solve for the calibration constant

theShowerCalib.SaveAlphas(); // Save the gain constants to use in the analysis

theShowerCalib.FillHEcal(); // Fill histograms
```
Calibration output

Calibration constants are saved in hcal.param.<RunNumber>, in format compatible with engine:

```
; Calibration constants for run 52949, 439596 events processed

hcal_pos_gain_cor= 0.430, 0.354, 0.416, 0.385, 0.247, 0.511, 0.550, 0.380, 0.501, 0.342, 0.385, 0.378, 0.000,
0.374, 0.424, 0.341, 0.386, 0.448, 0.484, 0.224, 0.448, 0.302, 0.272, 0.331, 0.638, 0.580,
0.604, 0.626, 0.602, 0.852, 0.628, 0.655, 0.465, 0.547, 0.713, 0.726, 0.522, 0.635, 0.000,
0.786, 0.603, 0.580, 0.731, 0.732, 0.828, 0.855, 0.742, 0.830, 0.625, 0.699, 0.738, 0.000,
hcal_neg_gain_cor= 0.419, 0.270, 0.287, 0.366, 0.353, 0.050, 0.185, 0.445, 0.236, 0.337, 0.203, 0.171, 0.000,
0.404, 0.357, 0.387, 0.510, 0.331, 0.337, 0.454, 0.301, 0.469, 0.361, 0.340, 0.380, 1.102,
0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000,
0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000,
```

*hcal_calib.cpp was checked against engine, for the same cuts difference in calibration constants is 0.001 or less.*
Graphics Output

E_{dep}/P uncalibrated

E_{dep}/P calibrated

\theta P versus E_{dep}/P
Difference in energy between golden track in hcana and best track in Engine

For comparisons, some modifications have been made in hcana:

In THcShowerCluster::clX method:

```c
// X coordinate of center of gravity of cluster, calculated as hit energy weighted average.
// Put X out of the calorimeter (-75 cm), if there is no en in the cluster.

Double_t clX() {
    Double_t x_sum=0.;
    Double_t Etot=0.;
    for (THcShowerHitIt it=THcShowerHitList::begin(); it!=THcShowerHitList::end(); ++it) {
        x_sum += (*it)->hitX() * (*it)->hitE();
        Etot += (*it)->hitE();
    }
    // Consistent with Engine.
    for (THcShowerHitIt it=THcShowerHitList::end()-1; it>=THcShowerHitList::begin(); it--) {
        x_sum += (*it)->hitX() * (*it)->hitE();
        Etot += (*it)->hitE();
    }
    return (float(x_sum) / float(Etot));
}
```

Make calculations in float precision, in the same order as in Engine
Modifications in hcana

In THcShower::MatchCluster method:

```c
// Since hits and clusters are in reverse order (with respect to Engine),
// search backwards to be consistent with Engine.
for (Int_t i=fNclust-1; i>0; i--)
{
    THcShowerCluster* cluster = (*fClusterList).ListedCluster(i);
    Double_t dx = TMath::Abs( (*cluster).clX() - XTrFront );
    if (dx <= (0.5*BlockThick[0] + fSlop)) {
        fNtracks++; // number of shower tracks (Consistent with engine)
        if (dx < deltaX) {
            mclust = i;
            deltaX = dx;
        }
    }
}
```

Search clusters for track association in the same order as in Engine.

In THcShower::IsNeighbour method:

```c
// Decide if a hit is neighbouring the current hit.
// Two hits are neighbours if share a side or a corner.

bool isNeighbour(THcShowerHit* hit1) {
    Int_t dRow = fRow-(*hit1).fRow;
    Int_t dCol = fCol-(*hit1).fCol;
    return (TMath::Abs(dRow)<2 && TMath::Abs(dCol)<2 ||
            (dRow==0 && TMath::Abs(dCol)<3));
}
```

Combine isolated small energy hits (from noisy channels) with big clusters.
hcana – Engine (golden track – best track) comparison

Run 52948, full replay, total number of analyzed events ~600K. *Same* golden/best tracks are compared. *The beta cut is not used.* **Difference in energies is mostly due to rounding errors.**
Summary

• Calibration for HMS calorimeter in hcana is done:

  Two step process – replay the run and calibrate the
  replayed run

  The Obtained calibration constants match the Engine
  results

• hcana – Engine (golden track – best track)
  comparison shows minor differences - mostly caused by
  rounding errors